ARCHITECTURE DES MICROPROCESSEURS

Baccalauréat S-Spécialité ISN

- 4.4 : Architecture matérielles : Structure des ordinateurs
  - **Éléments d'architecture**
    Composants de base (unité centrale, mémoires, périphériques).
  - **Jeu d'instructions**
    Instructions simples (chargement, stockage, opérations arithmétiques et logiques, saut conditionnel).

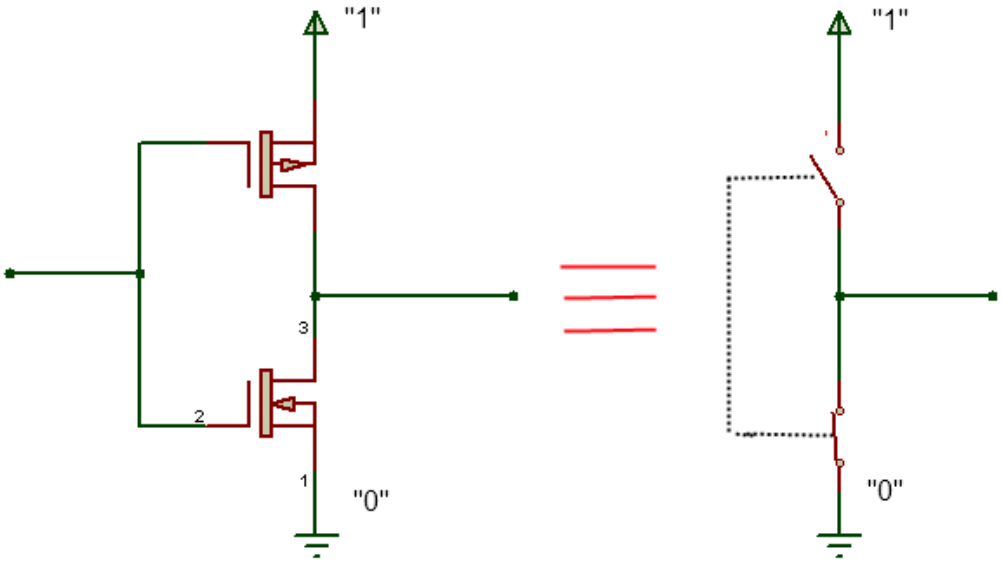
Rappels

Transistors et logique binaire

Les structures de l'électronique numérique reposent toutes sur des composants appelés ***transistors***.

Les transistors s'apparentent à des interrupteurs qui laissent passer ou non le courant électrique. L'utilisation de transistors complémentaires (quand l'un laisse passer le courant, l'autre le bloque) permet d'obtenir des cellules unitaires qui délivrent soit un niveau logique "0" ou un niveau logique "1". Ce fonctionnement binaire - on parle de logique

binaire -explique le fait que l'on utilise la base 2 pour coder des programmes informatiques.



Les microprocesseurs renferment des millions de transistors qui, lors de la commutation, provoquent un léger échauffement thermique. Ce phénomène multiplié des milliers de fois par seconde et par des millions de transistors explique l'impérieuse nécessite de refroidir ces circuits intégrés.

Bit et octet

Le bit :

De l'anglais binary digit, le bit décrit une variable n'ayant que deux états possibles.
Par exemple : Un nombre peut être pair ou non.

Lorsque des combinaisons intermédiaires sont nécessaires, on associe plusieurs bits pour obtenir des variables de valeurs plus grandes.

Un peu d'histoire....

Le premier microprocesseur, le 4004 d'INTEL était capable de traiter des données de 4 bits (un quartet).

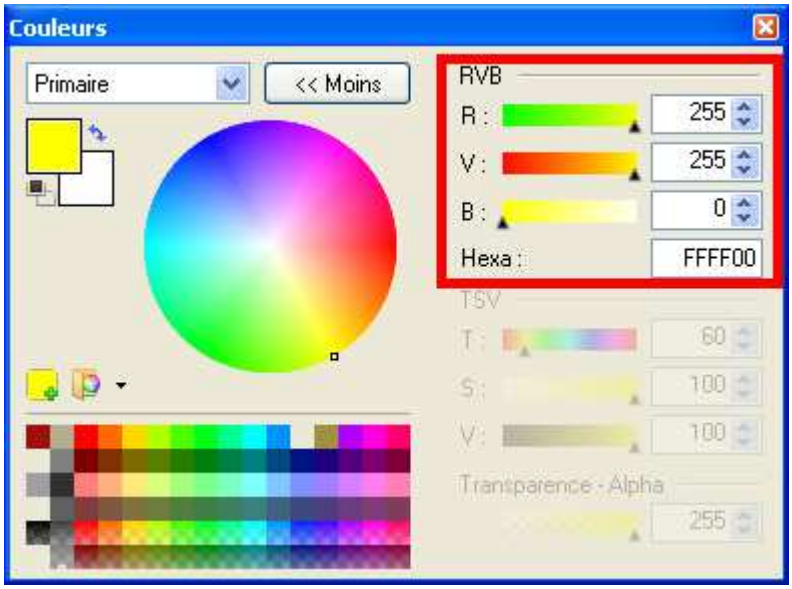
Avec un quartet, une variable pouvait avoir 16 valeurs différents (2 à la puissance 4).

Pour représenter ces valeurs sur un caractère, on a fait appel au code hexadécimal.

Mais très vite, on s'est aperçu que le choix de mots de 4 bits était très limitatif tant en nombre de valeurs, qu'en temps de calcul.

Les microprocesseurs qui ont suivi le 4004 (6800 de Motorola, Z80 de Zilog, 6805 de Fairchild, 8008 d'Intel....) furent dotés de calculateurs de 8 bits, multipliant par la même occasion les temps de calcul par 2 ou 3.

L'octet



Avec un octet - 8 bits - une valeur peut représenter 256 valeurs différentes (2 à la puissance 8).

Par exemple la couleur d'un pixel à l'écran est caractérisée par 3 variables qui sont ROUGE, VERT et BLEU. Ainsi la couleur jaune est définie par ROUGE=255=$FF VERT=$FF=255 et BLEU = $00=0

Remarque : pour signaler un mot hexadécimal, on utilise la lettre $ ou h ou 0x.

Exemple $FF=FFh=0xFF

Structures combinatoires

On appelle structure combinatoire, l'association d'éléments logiques pour lesquels un changement d'une variable d'entrée introduit immédiatement un changement possible de la sortie.

Exemple :

Soit un coffre fort disposant de quatre boutons. Si chaque bouton est positionné sur la bonne valeur, le coffre s'ouvre. L'ordre de manipulation n'a aucune importance. L'ouverture est simplement conditionnée par la bonne combinaison.

Il s'agit d'un fonctionnement combinatoire.

Les portes logiques sont des structures combinatoires.

Structures séquentielles

On appelle structure séquentielle, l'association d'éléments logiques faisant intervenir la notion de mémoire et d'états précédents pour provoquer le changement d'état de sortie de la structure.

Exemple :

Une carte bancaire dispose d'un code secret de 4 chiffres. L'ordre de saisi à son importance. Pour que le code soit accepté, il faut que les 4 numéros soient exacts et saisis dans l'ordre.
Il s'agit d'un fonctionnement qui respecte une séquence. Il s'agit d'un fonctionnement séquentiel.

Les registres, les bascules logiques, les compteurs sont des structures séquentielles.

Etude du système de traitement

Le rôle d'un système de traitement est de traitement de manière séquentiel les instructions d'un programme.
On rappelle que :

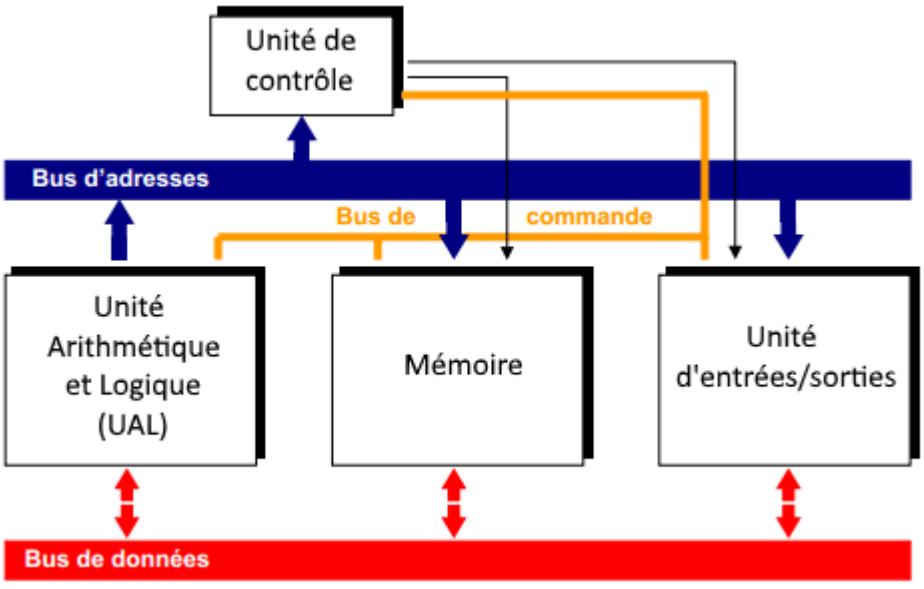
PROGRAMME = ALGORITHME + DONNEES

A l'aube de l'informatique, en 1945 le mathématicien Von Neuman, à la tête d'une équipe de chercheurs américains (Eckert et Mauchly en faisaient parti) a donné son nom à une arc hitecture de calcul, elle même basée sur les travaux d'Alan Turing.

L'architecture de Von Neumann

Dans le modèle de Von Neumann, 4 parties composent l'ordinateur :

- L'***unité arithmétique et logique (UAL)*** qui effectue les opérations arithmétiques et logiques
- **L'unité de contrôle** qui organise et séquence les opérations
- **La mémoire** qui contient les programmes ET les données
- **L'unité d'entrées/sorties** qui permet de communiquer avec le monde extérieur.

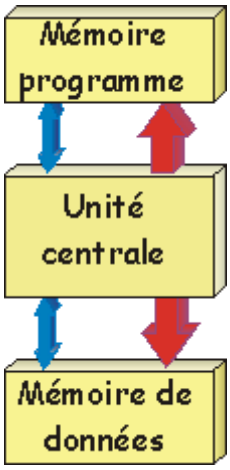


Les données (codes de programme et dopnnées) sont véhiculées par un ensemble de fils appelé ***BUS DE DONNEES***. Le bus de données est bidirectionnel et son nombre de fil dépend de la capacité de traitement du microprocesseur. 8 fils s'il s'agit d'un microprocesseur 8 bits.

Le bus d'adresse, unidirectionnel, véhicule le numéro de la case mémoire vers laquelle la donnée doit aller, ou d'où elle doit venir.

Le bus de commande comporte des signaux utiles au fonctionnement de l'ensemble (signaux de cadencement, sélection de lecture ou d'écriture, etc....).

Architecture Harvard



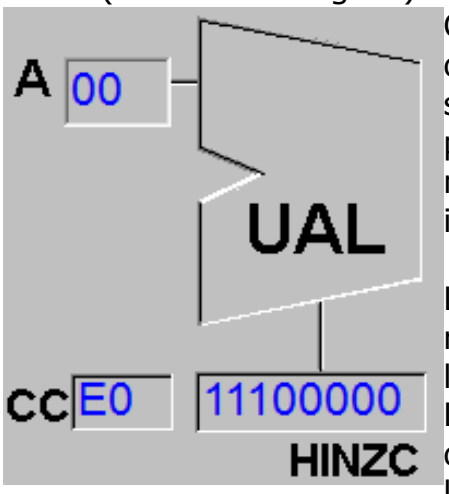
En même temps que Von Neumann menait ses travaux, l'université d'Harvard développait une architecture légèrement différente où données et instructions se trouvent dans des mémoires différentes ce qui permet un accès simultané, et donc de meilleures performances en vitesse mais pour un coût plus élevé et une plus grande complexité.

Cette architecture ne sera pas étudiée dans ce cours.

Architecture Von Neumann

L'unité Arithmétique et Logique

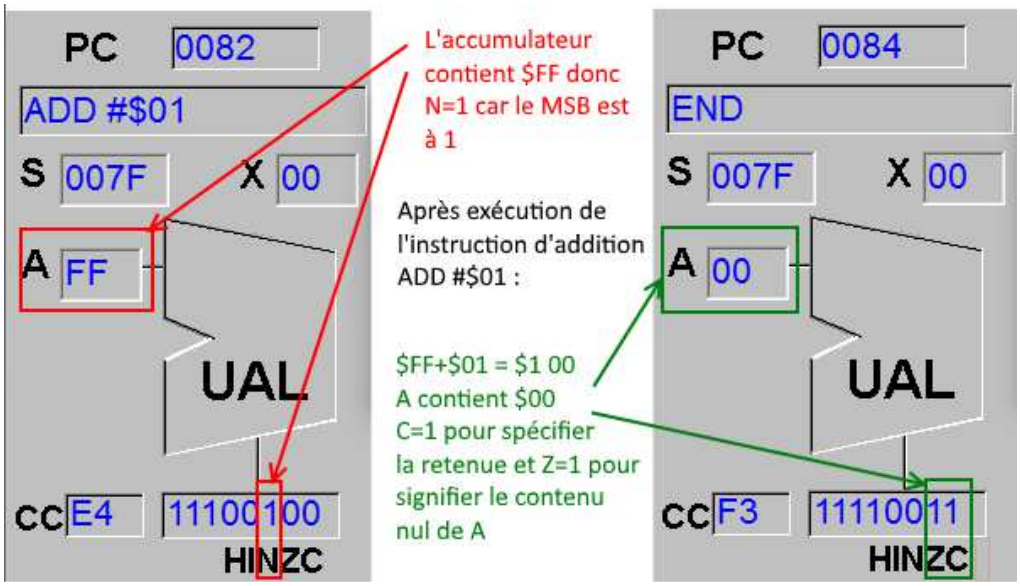
L'UAL (ou ALU en anglais) est la partie principale du microprocesseur.



C'est elle qui réalise les opérations et il s'agit d'une structure combinatoire dont la succession des opérations est commandée par l'unité de contrôle. Selon le modèle, le nombre d'opérations peut être plus ou moins important.

L'UAL est lié à des mémoire de taille très réduite appelées registres qui permettent à l'ual de pouvoir fonctionner.
L'accumulateur (parfois il y en a deux comme dans le 68HC11) mémorise l'opérande à l'entrée de l'UAL et sert aussi à stocker le résultat de l'opération.
Le drapeau ou registre de Code Condition donne des informations sur le résultat d'une opération (nul, négatif, dépassement de capacité, etc...)

Exemple de $FF+$01 (Simulation de l'instruction ADDA avec MOTO6805):



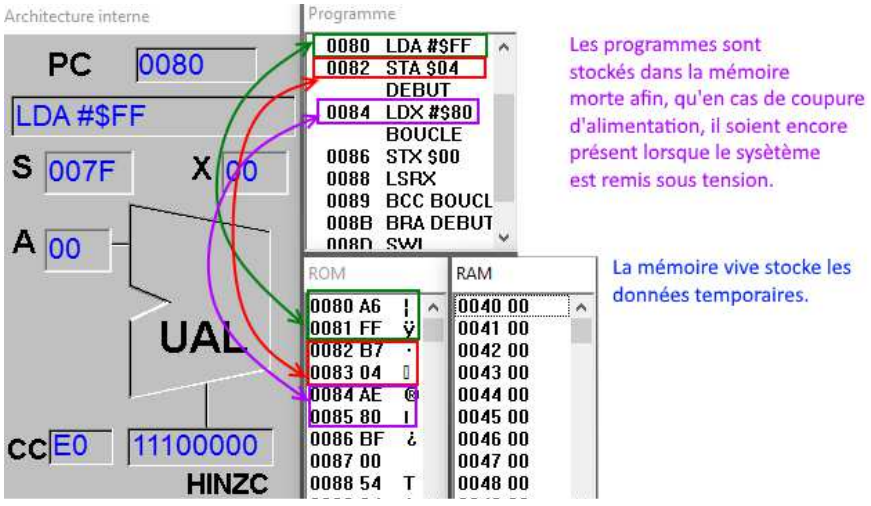
La mémoire

Elle contient les instructions du ou des programmes en cours d'exécution et les données associées à ce programme. Physiquement, elle se décompose souvent en :

- **une mémoire morte** (***ROM*** = Read Only Memory) chargée de stocker le programme. C'est une mémoire à lecture seule.
- **une mémoire vive** (***RAM*** = Random Access Memory) chargée de stocker les données intermédiaires ou les résultats de calculs. On peut la lire ou l'écrire, mais ses données sont perdues à la mise hors tension.

Chaque cellule mémoire comporte une "adresse" unique qui lui est propre.

Cette adresse est véhiculée par le bus d'adresse tandis que la donnée est véhiculée sur le bus de données.



Remarque :

Les disques durs, disquettes, CDROM, etc… sont des périphériques de stockage et sont considérés comme des mémoires secondaires. Elles nécessitent un contrôleur de disque.

L'unité d'entrées-sorties

Elle assure la communication entre le microprocesseur et les périphériques. (capteur, clavier, moniteur ou afficheur, imprimante, modem, etc…).

Elle est associée à des composants appelés COUPLEURS.

Les différents registres de ces coupleurs se comportent comme des cellules mémoires et possèdent de ce fait des adresses et des données qui précisent le comportement des entrées-sorties.

L'unité de contrôle

L'unité de contrôle décode les instructions et assure le séquencement des opérations et en particulier le passage des données. Elle se comporte un peu comme un programmateur de lave-linge.

Elle est composée :

- du **registre d'instruction** qui mémorise le code de l'instruction à exécuter
- le **décodeur d'instruction** qui décode l'instruction pour uq le séquenceur puisse organiser le séquencement des tâches
- le **séquenceur**.

L'unité de contrôle a besoin du compteur de programme (PC) qui lui indique quelle est la prochaine instruction à exécuter.

Système minimum

Système minimum ou micro-contrôleur

Le microprocesseur tout seul ne sert à rien. Il a besoin d'un environnement matériel et logiciel pour répondre à un besoin.

- De la mémoire
- Des périphériques d'entrées-sorties :
  - temporisateur programmable
  - coupleur de transmission série et parallèles
  - convertisseurs Analogiques-Numériques et Numériques-Analogiques
- Des programmes

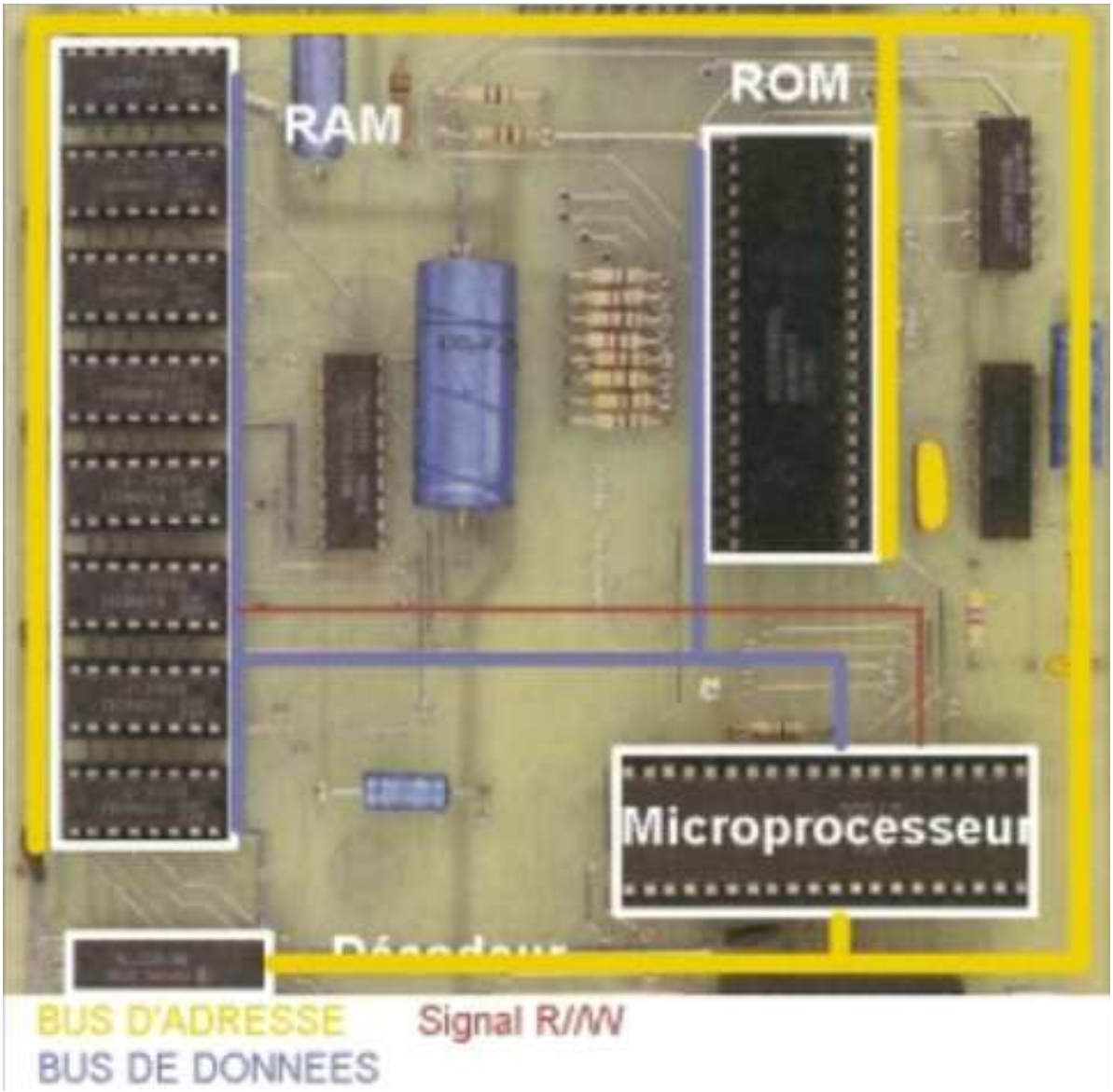
Lorsque qu'un même circuit intégré possède ces structures on parle de ***micro-contrôleur***.

Décodage d'adresse

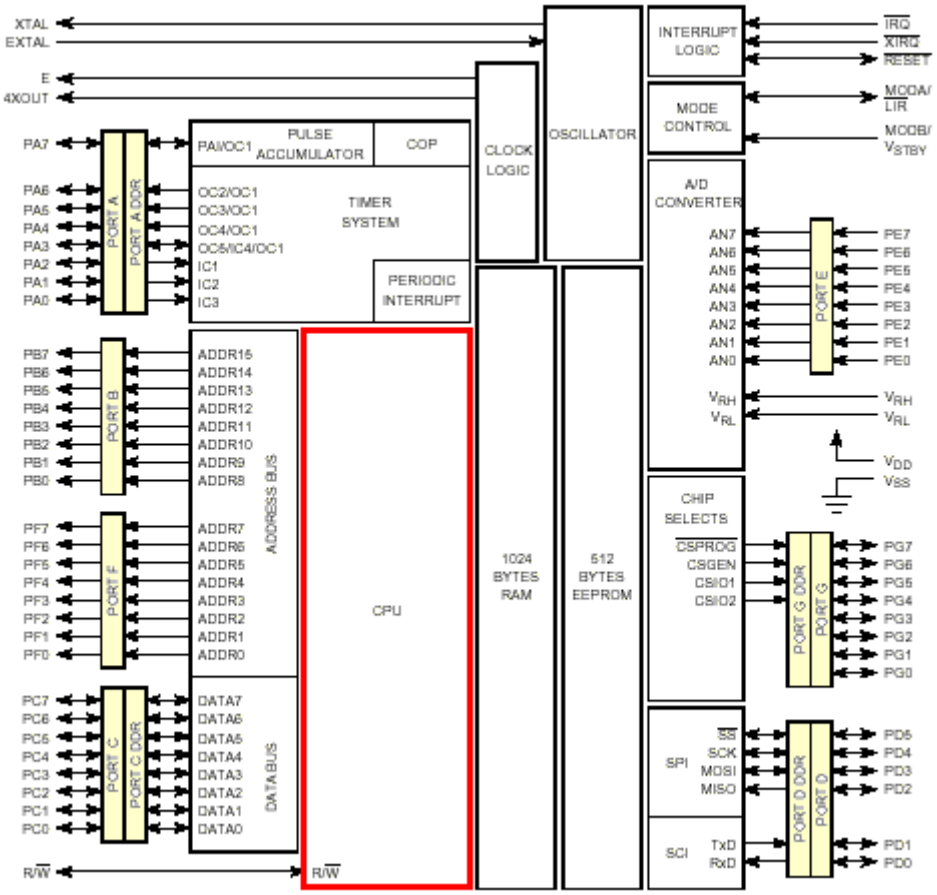
La multiplication des périphériques autour du microprocesseur oblige la présence d'un décodeur d'adresse chargé d'aiguiller les données présentes sur le bus de données.

En effet, le microprocesseur peut communiquer avec les différentes mémoires et les différents boîtier d'interface. Ceux-ci sont tous reliés sur le même bus de données et afin d'éviter des conflits, un seul composant doit être sélectionné à la fois.
Lorsqu'on réalise un système micro-programmé, on attribue donc à chaque périphérique une zone d'adresse et une fonction « décodage d'adresse » est donc nécessaire afin de fournir les signaux de sélection de chacun des composants.
Ces signaux se nomment généralement Chip Select (CS).

Exemple de système minimum



Exemple de la structure interne d'un 68HC11F1



Evolution des microprocesseurs

Les microprocesseurs ne cessent d'évoluer tant en terme de

- vitesse d'exécution,
- de consommation
- et d'intégration.

Le tableau ci-dessous présente l'évolution de 1971 à 2010 :

Date	Nom	Nombre de transistors	Finesse de gravure (µm)	Fréquence de l'horloge	Largeur des données	MIPS
1971	4004	2 300		108 kHz	4 bits/4 bits bus	
1974	8080	6 000	6	2 MHz	8 bits/8 bits bus	0,64
1979	8088	29 000	3	5 MHz	16 bits/8 bits bus	0,33
1982	80286	134 000	1,5	6 à 16 MHz (20 MHz chez AMD)	16 bits/16 bits bus	1
1985	80386	275 000	1,5	16 à 40 MHz	32 bits/32 bits bus	5
1989	80486	1 200 000	1	16 à 100 MHz	32 bits/32 bits bus	20
1993	Pentium	3 100 000	0,8 à 0,28	60 à 233 MHz	32 bits/64 bits bus	100
1997	Pentium II	7 500 000	0,35 à 0,25	233 à 450 MHz	32 bits/64 bits bus	300
1999	Pentium III	9 500 000	0,25 à 0,13	450 à 1 400 MHz	32 bits/64 bits bus	510
2000	Pentium 4	42 000 000	0,18 à 0,065	1,3 à 3,8 GHz	32 bits/64 bits bus	1 700
2004	Pentium 4D « Prescott »	125 000 000	0,09 à 0,065	2.66 à 3,6 GHz	32 bits/64 bits bus	9 000
2006	Core 2™ Duo	291 000 000	0,065	2,4 GHz (E6600)	64 bits/64 bits bus	22 000
2007	Core 2™ Quad	2*291 000 000	0,065	3 GHz (Q6850)	64 bits/64 bits bus	2*22 000 (?)
2008	Core 2™ Duo (Penryn)	410 000 000	0,045	3,33 GHz (E8600)	64 bits/64 bits bus	~24 200
2008	Core 2™ Quad (Penryn)	2*410 000 000	0,045	3,2 GHz (QX9770)	64 bits/64 bits bus	~2*24 200
2008	Intel Core i7 (Nehalem)	731 000 000	0,045 (2008) 0,032 (2009)	2,66 GHz (Core i7 920) 3,33 GHz (Core i7 Ext. Ed. 975)	64 bits/64 bits bus	?
2009	Intel Core i5/i7 (Lynnfield)	774 000 000	0,045 (2009)	2,66 GHz (Core i5 750) 2,93 GHz (Core i7 870)	64 bits/64 bits bus	?
2010	Intel Core i7 (Gulftown)	1 170 000 000	0,032	3,33 GHz (Core i7 980X)	64 bits/64 bits bus	?

Aspects logiciels

Type d'instructions

Les instructions d'un microprocesseurs sont souvent nombreuses mais élémentaires.
Il existe différents types d'instructions :

- opérations arithmétiques ou logiques : elles sont en lien avec l'UAL : addition, complémentation...
- opérations d'échanges ou de transfert : sauver l'accumulateur, charger l'accumulateur avec une donnée....
- opérations liées au fonctionnement du processeur : arrêt logiciel, masquage d'interruption...

Dans certains cas, les opérations ne nécessitent pas de d'opérande et le code machine de l'instruction se limite à un octet. Exemple : incrémenter A (INCA). On parle d'***adressage implicite***.

D'autres instructions nécessitent un opérande de 1 octet spécifié immédiatement à la suite de l'instruction. Exemple : ajouter $20 à A (LDAA #$20 en assembleur 6811). Il s'agit d'un ***adressage immédiat***.

Pour d'autres instructions encore, l'opérande est spécifié par l'adresse mémoire dans laquelle il se trouve. Exemple : mettre le contenu de la case mémoire 1000 dans A (LDAA $1000). Il s'agit d'un ***adressage***

direct (adresse sur un octet) ou un ***adressage étendu*** (adresse sur 2 octets).

Comparaison : programmation en assembleur et en C

En C, toute variable doit être déclarée :

Par exemple : char monVar;

Cela permet de réserver un espace mémoire dans la mémoire.

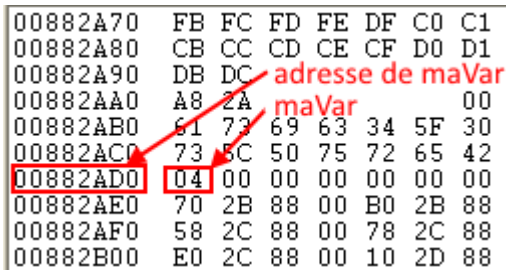
En assembleur, le programmeur doit faire l'effort de réserver ses cases mémoires lui même. Pour associer un nom à l'adresse de la case mémoire, il doit utiliser la pseudo instruction EQU.

Exemple :

```
PortA EQU $1000
DDRA EQU $1001
DDRG EQU $1003
PortG EQU $1002
```

En C, pour trouver, l'adresse de la case mémoire renfermant une variable, il faut utiliser le caractère &

Ainsi, ***&maVar*** rend l'adresse où se trouve la variable monChar.



Dans cet exemple, maVar=4 et &maVar=00882AD0

En assembleur, on précise l'adresse de rangement d'une valeur.

L'équivalent de maVar=4 serait si maVar est stockée en $1000 :

```
ldaa #$04
staa $1000
```

L'écriture par défaut (sans EQU) utilise donc directement l'adresse de stockage

Un pointeur est une variable qui contient l'adresse d'une variable

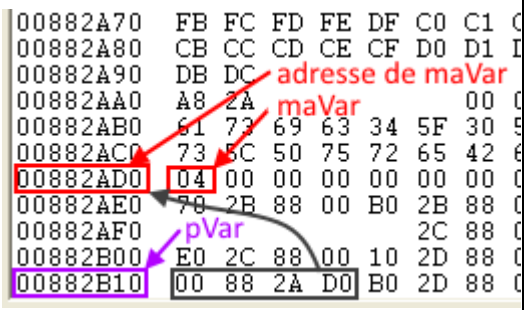
En C, la déclaration d'un pointeur se fait par :

char *pVar(0)

Exemple d'affectation de variable :

```
char temp;
pVar = &maVar;
temp = *pVar;
```

A la fin de l'exécution, temp=maVar



En assembleur, on dispose de registres d'index (X et parfois Y) qui contiennent l'adresse d'une case mémoire contenant la variable.

Exemple : la case mémoire $1000 contient $04

```
adVar EQU $1000
```

on affecte à X l'adresse de la case mémoire contenant $04

```
ldx #advar
ldaa ,x
```

A aura la valeur $04 à la fin de l'exécution